An Example Routine Calling the “C” Implementation of the Anderson-Moore(AIM) Algorithm

Gary S. Anderson and Rebecca Zarutskie
Board of Governors
Federal Reserve System
Washington, DC 20551
Voice: 202 452 2687
Fax: 202 728 5892
ganderson@frb.gov

August 19, 1998

Abstract

This paper describes how to use the “C” implementation of the Anderson-Moore Algorithm[1, 2, 3] for imposing the saddle point property in dynamic models. The paper uses a simple two-equation firm value model to demonstrate model construction and solution.

Contents

1 Introduction and Summary

This paper describes how to use the MATLAB implementation (currently downloadable at http://federalreserve.gov/pubs/oss/oss4/ code.html) of the Anderson-Moore Algorithm[1, 2, 3] for imposing the saddle point property in dynamic models. The paper uses a simple two-equation model to demonstrate model construction and solution.

2 The Firm Value Model

This paper uses AIM to investigate the solution of a simple linear model, first presented in [3], which describes the value of a firm.

The model consists of two equations:

$$\begin{aligned} V_{t+1} &= (1+r)V_t - D_{t+1} \\ D_t &= (1-\delta)D_{t-1} \end{aligned} \tag{1}$$

where V is the value of the firm, D is the dividend, r is the interest rate, δ is the growth rate of the dividend (here, negative).

3 Model Representation and Preprocessing

Describe the linear model using the MDLEZ syntax and save the model in a file, here called firmvalue.mdl

MODEL> Provides a name for the model. This does not affect the AIM calculations.

ENDOG> Provides names of the endogenous variables. In the AIM formulation, modellers must completely describe the long run behavior of the system. As a result, all variables are endogenous. "Exogenous" variables must have, at least, a simple forecasting equation.

EQUATION> Provides a name for the equation. This does not affect the AIM calculations.

EQTYPE> Specifies the type of equation. This does not affect the AIM calculations. The keyword STOCH indicates the equation has a stochastic error term. The keyword IMPOSED indicates the equation has no error term.

EQ> Provides the model equation.

"firmvalue.mdl" 3a ≡

```
MODEL> FIRMVALUE

ENDOG>
                    V
                    DIV


EQUATION> VALUE
EQTYPE> IMPOSED
EQ>     LEAD(V,1) = (1+R)*V - LEAD(DIV,1)

EQUATION> DIVIDEND
EQTYPE> IMPOSED
EQ>     DIV = (1-DELTA)*LAG(DIV,1)


END
```

◊

Run the model file through the model preprocessor and compile the resulting "C" program. The executable, mdlezAimC, is available on the Internet at http://www.federalreserve.gov/pubs/oss/oss4/parser.html.

```
mdlezAimC firmvalue.mdl > firmvalue.c
gcc -c firmvalue.c
```

This creates an executable file firmvalue.o.

4 Using the Anderson Moore Algorithm

Create a program to call AIM and the supporting routines in firmvalue.o The small numbers to the right of the descriptions refer to sections in the paper describing the "C" code.

"firmCaller.c" 3b ≡

```
⟨global declarations, include files and defines 5a⟩
main() {
⟨declare variables 5b⟩
⟨allocate model specification variables 6a⟩
⟨call procedure for obtaining model specification 6b⟩
⟨allocate matrices for aim results 6c⟩
⟨call procedure for constructing g and h given p 7⟩
⟨call aim and print some of the results 8⟩
}
```

◊

Compile the caller

```
gcc -c firmCaller.c
```

This creates the executable file firmCaller.o.
Link the programs with the AIM library.

```
f77 -o firmvalue firmvalue.o firmCaller.o $AIMLIB
```

where for $AIMLIB points to the AIM routines, LAPACK and BLAS routines, and the SRRIT routines.

```
AIMLIB  = -lfaim -lsrrit lapack_os5.a blas_os5.a
```

The archives faim.a srrit.a lapack_os5.a blas_os5.a are contained in the tar file of C libraries.

This creates the executable file firmvalue which produces the output:

```
Number of parameters: 2
Parameter names:
   R
   DELTA

G Matrix (transposed):
0.000  0.000
0.000  -0.100
-1.100  0.000
0.000  1.000


H Matrix (transposed):
0.000  0.000
0.000  0.000
0.000  0.000
0.000  0.000
1.000  0.000
1.000  0.000


roots, (1.100000,0.000000)
roots, (0.100000,0.000000)
roots, (0.000000,0.000000)
roots, (0.000000,0.000000)
q matrix has 2 rows.
q matrix has 1 rows associated with large roots.
q matrix has 1 rows associated with auxiliary initial conditions.
Caller has terminated normally with inform = 0.
```

5 Supporting Routines

⟨global declarations, include files and defines 5a⟩ ≡

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define NEGLIGIBLEDOUBLE 1.0e-9
#define UPPERLIM 1.0
void compute_aim_data();
void compute_aim_matrices();
```

◊

Macro referenced in scrap 3b.

⟨declare variables 5b⟩ ≡

```
  /*model specification parameters*/
  int np,neq,nlag,nlead;
  double *p;
  char ***param;
  char **modname;
  char ***eqname;
  int **eqtype;
  char ***endog;
  int **delay;
  int **vtype;
  double **g;
  double **h;

  /*aim parameters*/
  double cond;
  double epsi;
  double uprbnd;
  int iq,nexa,nnum,nbig,ndim;
  double *rootr,*rooti,*cof,*cofq;
  int *nroot,itsbad,inform;

  /*miscellaneous variables for loops counters*/
  int i,j;
```

◊

Macro referenced in scrap 3b.

⟨allocate model specification variables 6a⟩ ≡

```
param = (char ***) calloc(1,sizeof(char**));
modname = (char **) calloc(1,sizeof(char*));
eqname = (char ***) calloc(1,sizeof(char**));
eqtype = (int **) calloc(1,sizeof(int*));
endog = (char ***) calloc(1,sizeof(char**));
delay = (int **) calloc(1,sizeof(int*));
vtype = (int **) calloc(1,sizeof(int*));
g = (double **) calloc(1,sizeof(double*));
h = (double **) calloc(1,sizeof(double*));
```

◊

Macro referenced in scrap 3b.

⟨call procedure for obtaining model specification 6b⟩ ≡

```
compute_aim_data(param, &np, modname, &neq, &nlag, &nlead,
         eqname, eqtype, endog, delay, vtype);

printf("Number of parameters: %d\n", np);

printf("Parameter names:\n");
for (i = 0; i < np; i++)
  printf("   %s\n", (*param)[i]);
printf("\n");
```

◊

Macro referenced in scrap 3b.

⟨allocate matrices for aim results 6c⟩ ≡

```
/***  Change the following as needed for the particular problem  ***/
p = (double *) calloc(2, sizeof(double));  /* p = number of parameters */

ndim=neq*(nlag+nlead);
rootr=(double *) calloc(ndim*ndim,sizeof(double));
rooti=(double *) calloc(ndim*ndim,sizeof(double));
nroot=(int *) calloc(ndim*ndim,sizeof(int));
cof=(double *) calloc(neq*(neq+ndim),sizeof(double));
cofq=(double *) calloc(ndim*ndim,sizeof(double));
```

◊

Macro referenced in scrap 3b.

⟨call procedure for constructing g and h given p 7⟩ ≡

```
p[0] = 0.10;
p[1] = 0.90;


compute_aim_matrices(p, g, h);
printf("G Matrix (transposed):\n");
for (i = 0; i < (nlag + 1) * neq * neq; i++)
  {
    printf("%.3f  ", (*g)[i]);
    if (i % neq == neq - 1)
  printf("\n");
  }
printf("\n\n");

printf("H Matrix (transposed):\n");
for (i = 0; i < (nlag + 1 + nlead) * neq * neq; i++)
  {
    printf("%.3f  ", (*h)[i]);
    if(i < (nlag+ 1)*neq *neq)
      {cof[i]= (*g)[i] + (*h)[i];}
    else {cof[i] = (*h)[i];}
    if (i % neq == neq - 1)
  printf("\n");
  }
printf("\n\n");
```

◊

Macro referenced in scrap 3b.

⟨call aim and print some of the results 8⟩ ≡

```
    cond = NEGLIGIBLEDOUBLE ;
    epsi = NEGLIGIBLEDOUBLE ;
    uprbnd = UPPERLIM + NEGLIGIBLEDOUBLE;
    iq = 0;/*on input, the cofq array contains 0 auxiliary initial conditions*/

  faim_(cof,&neq,&nlag,&nlead,&epsi, &cond, &uprbnd, cofq,&iq,rootr, rooti,
       nroot,&nexa, &nnum, &nbig, &itsbad, &inform);
   for(i=0;i<ndim;i++){printf("roots,(%f,%f)\n",rootr[i],rooti[i]);}
  printf("q matrix has %d rows.\n",iq);
  printf("q matrix has %d rows associated with large roots.\n",nbig);
  printf("q matrix has %d rows associated with auxiliary initial conditions.\n",
  nnum+nexa);
  if(inform == 0)
  {  printf("Caller has terminated normally with inform =%d.\n",inform);}
  else {  printf("Caller has terminated with inform =%d.\n",inform);}
```

◊

Macro referenced in scrap 3b.

.

A Files

B Macros

C Identifiers

References

[1] Gary Anderson. A reliable and computationally efficient algorithm for imposing the saddle point property in dynamic models. Unpublished Manuscript, Board of Governors of the Federal Reserve System. Downloadable copies of this and other related papers at http://irmum1.frb.gov/˜m1gsa00/summariesAbstracts.html, 1997.

[2] Gary Anderson and George Moore. An efficient procedure for solving linear perfect foresight models. Unpublished Manuscript, Board of Governors of the Federal Reserve System. Downloadable copies of this and other related papers at http://irmum1.frb.gov/˜m1gsa00/summariesAbstracts.html, 1983.

[3] Gary Anderson and George Moore. A linear algebraic procedure for solving linear perfect foresight models. Economics Letters, 17, 1985.

[4] Michael W. Berry. Large scale sparse singular value computations. University of Tennessee, Department of Computer Science, 1996.

[5] Olivier Jean Blanchard and C. Kahn. The solution of linear difference models under rational expectations. Econometrica, 48, 1980.

[6] Gene H. Golub and Charles F. van Loan. Matrix Computations. Johns Hopkins, 1989.

[7] E. V. Krishnamurthy. Parallel Processing: Principles and Practice. Addison-Wesley, 1989.

[8] David G. Luenberger. Time-invariant descriptor systems. Automatica, 14:473–480, 1978.

[9] Ben Noble. Applied Linear Algebra. Prentice-Hall, Inc., 1969.

[10] J. Taylor. Conditions for unique solutions in stochastic macroeconomic models with rational expectations. Econometrica, 45:1377–1385, —SEP— 77.

[11] C. H. Whiteman. Linear Rational Expectations Models: A User's Guide. University of Minnesota, 1983.